МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Н.Э. Баумана

Факультет “Информатика и системы управления”  
Кафедра “Системы обработки информации и управления”

A logo with a gold star and blue ribbon

Description automatically generated

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по рубежному контролю №1  
Вариант Г10

**Выполнил:**Студент группы ИУ5-35Б  
Ким А. А.  
**Проверил:**Гапанюк Ю. Е.

подпись, дата

\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_

Москва 2025

**Задание**

В рамках рубежного контроля №1 по курсу «Парадигмы и конструкции языков программирования» требовалось разработать программу на языке Python.

**Цель работы:**

1. Создать два класса данных, соответствующие предметной области **«Браузер» (Класс 1) и «Компьютер» (Класс 2)** (Вариант №10).

2. Реализовать связь «один-ко-многим» между классами.

3. Создать дополнительный класс для реализации связи «многие-ко-многим».

4. Создать списки объектов классов, содержащие тестовые данные (3-5 записей).

5. Реализовать три запроса в соответствии с **Вариантом Г**, адаптировав их под выбранную предметную область.

6. При реализации запросов использовать функциональные возможности Python (list/dict comprehensions, функции высших порядков).

**Запросы:**

1. «Браузер» и «Компьютер» связаны соотношением «один-ко-многим». Вывести список всех компьютеров, у которых модель начинается с буквы «А», и список браузеров, установленных на них.

2. «Браузер» и «Компьютер» связаны соотношением «один-ко-многим». Вывести список компьютеров с максимальным использованием памяти браузеров на каждом компьютере, отсортированный по максимальному использованию памяти.

3. «Браузер» и «Компьютер» связаны соотношением «многие-ко-многим». Вывести список всех связанных браузеров и компьютеров, отсортированный по компьютерам.

**Листинг программы**

*# вариант 10 Г*from operator import itemgetter  
  
class Browser:  
 def \_\_init\_\_(self, id, name, memory\_usage, computer\_id):  
 self.id = id  
 self.name = name  
 self.memory\_usage = memory\_usage  
 self.computer\_id = computer\_id  
  
class Computer:  
 def \_\_init\_\_(self, id, model):  
 self.id = id  
 self.model = model  
  
class BrowserComputer:  
 def \_\_init\_\_(self, computer\_id, browser\_id):  
 self.computer\_id = computer\_id  
 self.browser\_id = browser\_id  
  
  
*# Компьютеры*computers = [  
 Computer(1, 'Asus ROG'),  
 Computer(2, 'Apple MacBook Pro'),  
 Computer(3, 'Dell XPS'),  
 Computer(11, 'Acer Predator'),  
 Computer(22, 'Apple iMac'),  
 Computer(33, 'Asus ZenBook'),  
]  
  
*# Браузеры*browsers = [  
 Browser(1, 'Chrome', 512, 1),  
 Browser(2, 'Firefox', 256, 2),  
 Browser(3, 'Safari', 384, 3),  
 Browser(4, 'Edge', 320, 3),  
 Browser(5, 'Opera', 192, 3),  
]  
  
*# Связи многие-ко-многим*browsers\_computers = [  
 BrowserComputer(1, 1),  
 BrowserComputer(2, 2),  
 BrowserComputer(3, 3),  
 BrowserComputer(3, 4),  
 BrowserComputer(3, 5),  
 BrowserComputer(11, 1),  
 BrowserComputer(22, 2),  
 BrowserComputer(33, 3),  
 BrowserComputer(33, 4),  
 BrowserComputer(33, 5),  
]  
  
def main():  
 *# Соединение данных один-ко-многим* one\_to\_many = [(b.name, b.memory\_usage, c.model)  
 for c in computers  
 for b in browsers  
 if b.computer\_id == c.id]  
  
 *# Соединение данных многие-ко-многим* many\_to\_many\_temp = [(c.model, bc.computer\_id, bc.browser\_id)  
 for c in computers  
 for bc in browsers\_computers  
 if c.id == bc.computer\_id]  
  
 many\_to\_many = [(b.name, b.memory\_usage, comp\_model)  
 for comp\_model, comp\_id, browser\_id in many\_to\_many\_temp  
 for b in browsers if b.id == browser\_id]  
  
 print('Задание Г1')  
 print('Список всех компьютеров, у которых модель начинается с буквы "A", и список браузеров на них:')  
  
 comps\_with\_a = list(filter(lambda i: i[2].startswith('A'), many\_to\_many))  
  
 *# Группируем по компьютерам* res\_1 = {}  
 for browser\_name, memory, comp\_model in comps\_with\_a:  
 if comp\_model not in res\_1:  
 res\_1[comp\_model] = []  
 *# Добавляем браузер, если его еще нет в списке для этого компьютера* if browser\_name not in res\_1[comp\_model]:  
 res\_1[comp\_model].append(browser\_name)  
  
 for comp, browsers\_list in sorted(res\_1.items()):  
 print(f'{comp}: {browsers\_list}')  
  
 print('\nЗадание Г2')  
 print('Список компьютеров с максимальным использованием памяти браузеров на каждом компьютере, отсортированный по максимальному использованию памяти:')  
  
 res\_2\_unsorted = []  
  
 *# Перебираем все компьютеры* for c in computers:  
 c\_browsers = list(filter(lambda i: i[2] == c.model, many\_to\_many))  
  
 if len(c\_browsers) > 0:  
 c\_memory = [memory for \_, memory, \_ in c\_browsers]  
 c\_memory\_max = max(c\_memory)  
 res\_2\_unsorted.append((c.model, c\_memory\_max))  
 else:  
 res\_2\_unsorted.append((c.model, 0))  
  
 *# Сортировка по максимальному использованию памяти (по убыванию)* res\_2 = sorted(res\_2\_unsorted, key=itemgetter(1), reverse=True)  
  
 for comp, max\_memory in res\_2:  
 print(f'{comp}: {max\_memory} МБ')  
  
 print('\nЗадание Г3')  
 print('Список всех связанных браузеров и компьютеров, отсортированный по компьютерам:')  
  
 *# Сортируем по названию компьютера* res\_3 = sorted(many\_to\_many, key=itemgetter(2))  
grouped\_result = {}  
 for browser\_name, memory, comp\_model in res\_3:  
 if comp\_model not in grouped\_result:  
 grouped\_result[comp\_model] = []  
 grouped\_result[comp\_model].append(f"{browser\_name} ({memory} МБ)")  
  
 for comp\_model, browsers\_list in sorted(grouped\_result.items()):  
 print(f'{comp\_model}: {browsers\_list}')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Скриншоты работы программы**

